

Limitations and Alternatives for the Evaluation of Large-scale Link Prediction

Garcia-Gasulla, D. & Ayguadé, E. & Labarta, J.
 Barcelona Supercomputing Center (BSC)
 dario.garcia@bsc.es

& Cortés, U.
 Universitat Politècnica de Catalunya - BarcelonaTECH

November 28, 2016

Abstract

Link prediction, the problem of identifying missing links among a set of inter-related data entities, is a popular field of research due to its application to graph-like domains. Producing consistent evaluations of the performance of the many link prediction algorithms being proposed can be challenging due to variable graph properties, such as size and density. In this paper we first discuss traditional data mining solutions which are applicable to link prediction evaluation, arguing about their capacity for producing faithful and useful evaluations. We also introduce an innovative modification to a traditional evaluation methodology with the goal of adapting it to the problem of evaluating link prediction algorithms when applied to large graphs, by tackling the problem of class imbalance. We empirically evaluate the proposed methodology and, building on these findings, make a case for its importance on the evaluation of large-scale graph processing. *Graph Mining Link Prediction Evaluation Methodology*

1 Introduction

The structural particularities of graphs (*i.e.*, networks) has motivated the design of specific machine learning methods for processing this type of data. These knowledge discovery tools typically try to exploit structural properties of high-dimensional, inter-connected data sets, with the goal of learning from the relational patterns of its entities. Among the names used to refer to some of these tools are:

- Graph-based data mining ([21, 46])
- Statistical Relational Learning ([17, 41])
- Link Mining ([16, 35])
- Network or Link Analysis ([19, 38])
- Network Science ([32])
- Structural Mining ([9])

For the sake of simplicity from now on we refer to all these methods using the general term *graph mining*.

The characterization of graph mining algorithms is relevant, not only because graphs represent data in a rather unique way, but because they are also able to capture a different type of information. While traditional table representations of entity/value pairs naturally capture *intra-entity patterns*, and so does traditional machine learning, network data captures mostly *inter-entity patterns*. Mining graphs therefore requires a shift in perspective, moving from an instance-attribute paradigm to an instance-instance paradigm.

These new methods of machine learning were designed to tackle network related problems such as:

- Finding the relevance of entities based on their relations or *link-based object ranking*[lu2016vital, 16] *e.g.*, PageRank ([42]) and HITS ([27])
- Finding groups of entities strongly related or *community detection*[fortunato2010community] *e.g.*, stochastic blockmodeling ([25])
- Finding reoccurring association patterns or *frequent subgraph discovery*[jiang2013survey], *e.g.*, Apriori based algorithms ([23])

These graph mining tasks, which have relations among entities as the cornerstone of their design, are applicable to domains such as life sciences ([3, 18, 45]), sociology and social networks ([1, 39, 48]), collaboration analysis ([2, 35, 43]), business and product recommendation ([6, 22]), and even law enforcement and anti-terrorism ([4, 8, 28]).

The increased dimensionality of graph data sets often comes hand in hand with an increase in size. Together, large dimensionality and size, define the increasingly frequent family of domains known as large scale networks (*e.g.*, Twitter, the brain connectome or web graphs). Regardless of the underlying domain, computing large scale networks represents a challenge in terms of efficiency, parallelism and scalability. Efficiency, because computing models and hardware architectures are not optimized for handling graph data. Parallelism, because the size of large networks makes serial approaches unfeasible. And scalability because limited computational resources constrain the applicability of exhaustive, model-based solutions. Beyond the challenges on *how* is the process implemented, the particularities of large scale networks also generate novel challenges on *what* must be done with the data from a data mining perspective. A prime example of that is deciding how to evaluate the performance of the mining algorithms in this novel setting.

In this paper we focus on the challenge of faithfully evaluating a graph mining task, Link Prediction (LP), when working with large scale graphs. The goal of LP is to find new or missing edges within a given graph. By using LP one can directly grow any data set represented as a graph using the same graph language (*i.e.*, adding edges among vertices by using only previous edges and vertices). As a result one could apply LP algorithms to virtually any domain that can be represented as a graph without supervision. The complexity of achieving good performance on the LP task increases with the graph size, as does the problems at faithfully evaluating performance. When a graph grows linearly in vertices, the number of possible links within the graph grows quadratically. This defines a *needle in a haystack* context where relevant or useful predictions are but a tiny fraction of all predictions. Keeping a good precision in this type of problem turns out to be very difficult, as the smallest false positive acceptance rate will amount to a huge absolute number of wrongfully predicted edges (*i.e.*, false positives). But in parallel, estimating the quality and applicability of results also becomes particularly difficult.

In §2 we explore the current solutions provided by the data mining community, particularly in the context of test set construction and class imbalance. We explore the features of those methods for the particular case of LP in §3, and argue on the utility of popular approaches like ten-fold cross validation and precision-recall curves. Then in §4 we propose an adapted evaluation methodology, and show an empirical analysis on its impact when applied to several graphs. Conclusions are presented in §6.

This paper is an extended version of [15], improving the definition of the proposed evaluation methodology, analysing its properties in more depth, and adding an empirical comparison between the proposed methodology and current solutions (§4). Further images and tables are provided to illustrate on the relevance of the contribution.

2 Evaluation context

LP and the rest of graph mining tasks represent a new family of data mining algorithms. The particularities of these algorithms originate from the special nature of networks. Particularities that include data dimensionality, variable dependency, and often log-scale distribution of information. Even with these differences, one can find analogies between graph mining problems and general data mining problems. From a traditional data mining perspective, LP can be reduced to a binary classification problem between two classes: the positive class of edges that do or should exist, and the negative class of edges that do not and should not exist. Given a directed graph $G = (N, E)$,

and all the possible edges in the graph (of size $|N * (N - 1)|$), the problem of LP would be that of distinguishing between those edges that exist, $e \in E$, and those that do not, $e \notin E$. The analogy between LP and binary classification is accurate in most cases, as the target of LP is often to identify the positive class. Which in terms of graph mining is equivalent to finding and proposing missing links. For the remaining of this paper we will assume this mainstream case.

In the bibliography there are a many methodologies available for the evaluation of a binary classification problem. These methodologies are typically discriminated based on the problem characteristics, which in the case of the LP classification problem are dominated by class imbalance.

2.1 Test Sets

To evaluate a binary classifier empirically we require a test set. Given a graph, LP algorithms can propose a number of edges to be added to it, however, to validate the quality of those proposals, we need a set of edges known to be correct and missing from the graph. In evaluation, each predicted edge found in the test set is considered as a correct prediction, while each predicted edge not found is considered as a mistake. From these results one can then obtain performance indicators like *precision* and *recall*.

The main problem with tests sets is how to obtain them. In the case of LP, the best test set one can use is that which represents a natural extension of the graph being computed. This is feasible on temporally grounded domains. For example, for a graph composed from Wikipedia articles and the hyperlinks among them from 2012, we can obtain a natural test set by considering the links added to Wikipedia after 2012 ([13]). Unfortunately, the domains and graphs having such incremental nature are rare. Instead, in most cases one must settle for the more drastic approach of randomly removing a number of edges from the graph in order to use them as test set. This yields other problems such as how to define a representative test set for the LP problem [zhu2012uncovering].

A frequent concern when one must split a set of data to produce a test set is representativeness. Typically, a random split cannot guarantee a prototypical distribution. The most frequent solution for avoiding bias is ten fold cross validation (10-fold CV). Within the LP problem, splitting data to build a test set will be often necessary ([50]). However, as we show in §3.2, performing 10-fold CV is redundant. Hence, for all the test performed in this article we will use a random split of 10% of edges on each graph to build the test set.

2.2 Class imbalance

A recurrent type of real world graphs are scale-free networks, from protein interactions, to social networks or the WWW [barabasi2009scale], a type of network where degree distribution follows a power-law. This distribution implies a significant sparsity in the graph [del2011all], which becomes more severe as the graph grows. It is indeed hard to find real world networks where the average vertex degree is over fifty ([2, 32, 34]), a feature consistent even as graphs grow to billions of vertices ([44]). In the context of reducing the LP problem to a binary classification problem, scale-free networks results in a severe class imbalance, as the negative class becomes much larger in comparison to the positive class. As is well known, class imbalance can be a severely complicating factor in classification problems ([7, 33, 47, 49]).

The degree of class imbalance found on large graphs when performing LP is hard to overestimate, and even for non-scale-free networks, large, dense graphs are very hard to come by. To illustrate on the type of class imbalance medium and large graphs may have, Table 1 shows the topological properties of some real world graphs obtained from WordNet ([14]), the Cyc project ([31]), the movie-related IMDb knowledge base ([13]), and several web graphs from the Notre-Dame University ([5]), Stanford/Berkley universities ([26]), a Google challenge ([13]), and the Hudong, Baidu ([29]) and Wikipedia encyclopedias ([30]). Notice how, in the *best* case scenario, the class ratio is of 1 positive instance for every 11,382 negative instances.

The impact of class imbalance on classifiers was explored in [24], and authors concluded that this impact was largely reduced when all classes were of reasonable size. *A priori* this should be good news for LP on large graphs, as its classes seem to be of *reasonable* size; the positive class of all graphs shown in Table 1 is over 10,000 entities. Unfortunately, this assumption does not apply to the LP problem ([32]), and the reason for this is twofold. On one hand the imbalance found in LP on large graphs is several orders of magnitude larger than any imbalance tested in [24].

Table 1: Sample of average number of edges per vertex and class imbalance on real graphs

Data source	Number of vertices	Average edges per vertex	positive:negative class ratio
WordNet	89,178	15.66	1:11,382
Cyc	116,835	5.9	1:39,496
webND	325,729	9.18	1:70,867
webSB	685,230	22.18	1:61,775
webGL	875,713	11.64	1:150,217
hudong	1,984,484	14.98	1:264,848
baidu	2,141,300	16.72	1:257,667
IMDb	2,930,634	5.12	1:1,140,835
DBpedia	17,170,894	19.44	1:2,151,672

Thus its impact may become significant at some point. On the other hand, LP is not a standard data mining classification problem, and given the small amount of information provided by each edge (*e.g.*, positive instances have no attributes), a class composed 30,000 elements could still be considered to be small. In reality, class imbalances of 1:10,000 or larger translate as a strong tendency towards false positive classification mistakes, as incorrectly accepting negative instances becomes almost inevitable. The main challenge of LP is therefore precision, a notion that should be taken into account by the evaluating methodologies.

2.3 Evaluation under class imbalance

Class imbalance is key in classification problems as it implies difficulties in predicting the small class. A small class that is in most cases the main target of the predictive process. Consequently there is a large and growing state-of-the-art on how to deal with class imbalance. A frequent approach of supervised or semi-supervised learning methods to overcome class imbalance is to equilibrate the training set through over-sampling, under-sampling or feature selection ([7, 33, 47, 49]). Unsupervised LP algorithms cannot benefit from these solutions as adding or removing edges from the data set would equal to perform arbitrary classification, and there are no features to be removed beyond the existence of edges among vertices. As a result, for the LP problem one must focus only on those aspects of class imbalance that are relevant for unsupervised methods: deciding which metrics to use when evaluating and comparing the performance of binary classifiers for data sets with a large class imbalance.

The most frequently used methods for classifier evaluation are based on accuracy. However, these methods are biased towards the classification of instances within the large class, making them inappropriate for imbalanced data sets ([7, 20, 33, 47]). Using them for LP would be almost analogous to measuring the capability of algorithms at predicting which edges should not be added to the graph, which is not the goal of LP. For data sets with large class imbalance, the most frequently used methodology is the Receiver Operating Characteristic (ROC) curve and the derived Area Under the Curve (AUC) measure ([11]). The ROC curve sets the True Positive Rate (TPR) against the False Positive Rate (FPR), making this metric unbiased towards entities of any class regardless of their size. The AUC measures the area below the curve in order to compare the overall predictive performance of two different curves.

ROC curves are unbiased in imbalanced contexts, but their consideration of miss-classifications can result in mistakenly optimistic interpretations ([10, 50]). When the negative class is very large, showing mistakes as relative to the negative class size (*i.e.*, FPR) can hide their actual magnitude, and make it complicated to assess the overall performance quality. From a practical perspective, most of the ROC curve is irrelevant when dealing with large class imbalance, as it represents completely unacceptable precisions. For example, one may consider that a classifier achieving a TPR of 0.95 (finding 95% of all positive edges) and a FPR of 0.01 (incorrectly accepting 1% of all negative edges) in the ROC curve demonstrates an excellent performance. However, for a data set with a positive:negative rate of 1:100 those results imply that the classifier accepts more negative edges than positive edges (*i.e.*, it has a precision smaller than 0.5). For domains with a 1:11,000 or worse ratio, like the ones shown in Table 1, the limitations of the ROC curve become even more striking. In those

even a FPR of 0.0001 implies a very poor precision/performance regardless of the TPR achieved.

Consider a theoretical graph defined by $N=100,000$ and $E=1,000,000$, for which we build a test set using 10% of the available edges. The positive class size of this graph will be 100,000, the negative class size 9,998.9 million, and the imbalance ratio 1:99,989. Notice this graph is not particularly imbalanced (see Table 1 for comparison with graphs coming from real-world domains). A ROC curve for a LP algorithm on this theoretical graph could look like the one shown in Figure1. A FPR of 0.1 for such a graph (incorrectly accepting 10% of negative class instances) would imply the wrong prediction of 999,890,000 edges, while our graph originally had 1,000,000 edges (900,000 after the test set split). Even with a FPR of 0.0001 a classifier would be making more false predictions than edges found in the graph. This simple example shows how most of the ROC curve is virtually useless for domains with a very large imbalance, which leads us to seriously question the utility of the associated AUROC measure in this context.

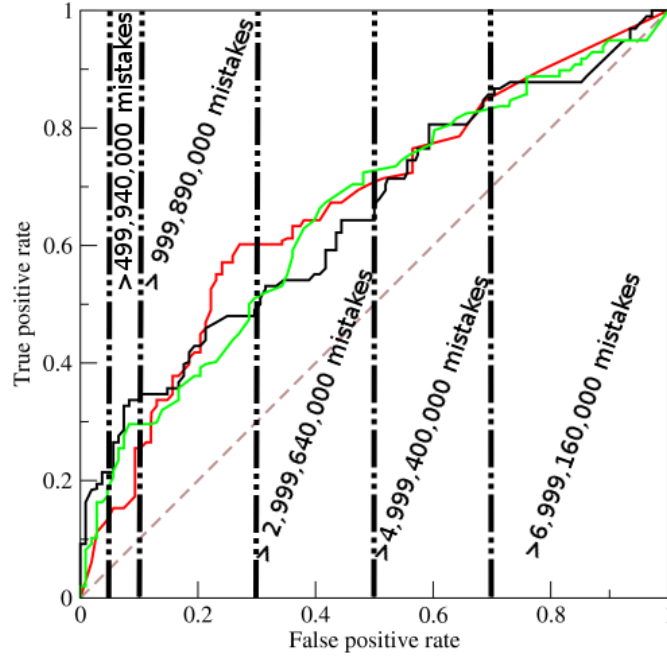


Figure 1: Example on the impact of imbalance on a ROC curve, showing the number of classification errors done at each FPR for a graph defined by $N=100,000$ and $E=1,000,000$. The true positive class size for this problem is 100,000.

Precision-recall (PR) curves are an alternative to ROC curves. A PR curve is resistant to class imbalances as it focuses only on the performance achieved for the positive class (typically the small one), and does not show the number of correct classifications for the negative class. PR curves plot precision (y axis) against recall (x axis), directly showing the precision of the classifier. Precision can also be obtained from ROC curves, but is not straightforwardly visible. ROC and PR curves are related; a curve dominates another (it is above it) in the ROC space if and only if it also dominates it in PR space, but are not equivalent; a classifier that optimizes on ROC space will not necessarily also optimize on PR space ([10]). One particularly relevant difference between ROC and PR curves regarding the interpretation of predictive performance is on how errors are represented. While ROC curves show miss-classifications as relative to the total number of negative cases, PR curves show miss-classifications as relative to the total number of predictions done.

An illustration on the impact of these differences is the curves defined by a random classifier, which always performs poorly in an imbalanced data set. The ROC curve always represents the random classifier as a straight line between points (0, 0) and (1, 1), regardless of class imbalance, with all better than random classifiers represented as lines above that diagonal. PR curves on the other hand represent random classifiers in imbalanced data sets a flat line on the x axis, as their precision in imbalanced settings is always close to zero. This alone shows that PR curves can provide richer characterizations of classifiers for imbalanced data sets.

3 Evaluating link prediction

Current solutions for performance evaluation, like the ones shown in §2, have severe limitations when applied to large graph mining problems. Issues like test set representativeness, or the evaluation under class imbalance, reach a new degree of relevance when considering problems like LP on large networks. In this section we discuss these problems in depth and propose solutions fitting our LP problem.

3.1 Representativeness of test sets

The use of a test sets to evaluate LP implies the assumption that the test set (the prediction of which is evaluated by the curves) faithfully represents the *correct* edges missing from the graph. Or in other words, that all edges not found in neither the graph nor in the test set, are wrong. In certain cases, where the graph topology is stable, this may be an accurate assessment. For example, a graph obtained from WordNet data (as shown in [14]) can be considered as almost perfect, since WordNet relations have been identified, discussed and implemented by linguists for decades. In other cases though test sets are an imperfect measure of the right edges missing from the graph. Consider for example a graph obtained from Wikipedia articles and hyperlinks, in which the pagelinks among Wikipedia articles from 2012 are used as training and the new pagelinks added on 2013 are used as test. This graph is clearly incomplete, as new links are being added every day. The Wikipedia grows continuously and the fact that a link is not implemented so far does not mean it is wrong. As a result, one must take into account that some of the edges predicted, not found in the test set and labelled as mistakes, will in fact be correct predictions corresponding to edges not yet added to the graph.

Using a test set which does not fully represent the target class implies an underestimation of performance, as the predictions being made outside of the test set will always (and not always correctly) be considered as mistakes. Nevertheless, since this limitation applies to all the methods being evaluated (assuming all methods are evaluated using the same test set), it can be argued that the resultant performance indicators remain valid for comparative purposes. That is, we can still find out which LP algorithm works better. The unavoidable shortcoming of representativeness comes when evaluating the precision of a score in the context of applicability. That is, we cannot be sure of how well performs the best LP algorithm. The only way to obtain a faithful, non-comparative evaluation of performance of a single LP algorithm would be a hand-made validation. One could achieve an approximate solution by performing a sampling process of all edges predicted, manually evaluating the sampled edges as correct or incorrect predictions, and then extrapolating the performance obtained on the sample to the rest of the graph. There are several aspects to keep in mind with this solution. First of all, the sampling needs to be large for the extrapolation to be faithful, which equals to many hours of manual labelling. And second, the sampling would have to be done at several thresholds so that extrapolations are representative of the whole curve. Sampling may therefore be the only accurate evaluation methodology for estimating predictive performance of a given score on a specific domain, at the price of a huge amount of manual labelling hours.

3.2 10-fold CV

10-fold CV is a commonly used technique for reducing variance in test set construction and improving representativeness. Although 10-fold CV is almost universally expected when using test sets that are a random portion of a complete data set, we argue that it is not needed when performing large graph mining. The main reason behind that argument being the large size of these domains, which naturally avoid variance. To assess the utility of 10-fold CV we test a webgraph obtained from a Google challenge, composed by 875,713 vertices and 5,105,039 edges. This particular graph could be considered to be medium sized, as it is easy to find much larger ones (see Table 1). The conclusions obtained for this graph could be extended, even with more reliability, to larger graphs.

A random 10% test set of the Google challenge webgraph is composed by 510,503 edges. We obtain ten different random test sets from this graph, and use each one of them to evaluate seven different LP algorithms. As a result we obtain ten PR curves (as these are preferred to ROC curves, see §2.3), for seven different LP algorithms. Algorithms used are RA (Resource Allocation), AA (Adamic-Adar), CN (Common Neighbours), INF, INF_LOG, INF_2D and INF_LOG_2D, all of which are described in [13]. In Figure 2 we show the ten curves belonging to one of those algorithms (INF_LOG_2D),

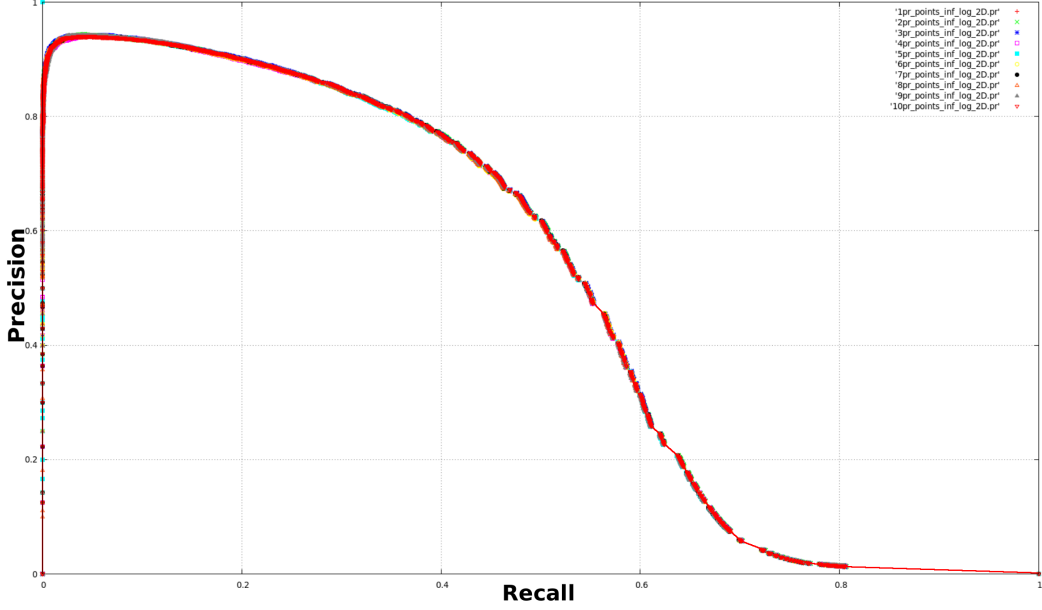


Figure 2: Ten precision-recall curves for the INF_LOG_2D LP algorithm when applied to 10 random splits of the Google challenge webgraph. The ten curves are clearly overlapped, showing minimal variance among random splits.

illustrating the minimal variance found among curves. The ten curves are virtually identical, which implies that variance among random splits is irrelevant. To empirically validate this assertion, in Table 2 we show the AUPR of the seventy curves obtained, ten for each algorithm. Results show that the variance of 10 executions using 10 randomly selected tests sets is very low. In fact, the standard deviation represents a 0.32% of the mean value in the worst case (algorithm #3). Such a low variance is the result of having a large test set, which, given the law of large numbers, will tend towards a stable sample. In this context it seems clear that performing 10-fold CV is not necessary, as a single run is a representative and accurate sample of performance.

Table 2: Using the Google challenge webgraph, AUPR obtained by seven different algorithms on ten different and randomly split test sets. The minimum and maximum value among the ten splits, and the standard deviation.

Algorithm	Min. AUPR	Max. AUPR	Mean	Std. Dev.
AA	0.0892558	0.0899991	0.08971357	0.0002287783
CN	0.10017	0.10083	0.1005145	0.0001902058
RA	0.0618143	0.0625483	0.06225763	0.0002040158
INF	0.128201	0.128857	0.1285072	0.0001879318
INF_LOG	0.124934	0.125385	0.1251525	0.0001210622
INF_2D	0.419577	0.421239	0.4204078	0.0004921798
INF_LOG_2D	0.491902	0.4935	0.4925985	0.0005283041

Regardless of these results, performing 10-fold CV is not a wrong or misleading strategy. Our argument here is that 10-fold CV is not required in order to consider some results representative. This fact is particularly relevant due to the computational cost of computing large scale graphs. Building test sets and running graph mining algorithms on them is typically expensive in computational terms. Hence, the physical resources and time spent doing ten equivalent executions could be use more efficiently elsewhere.

3.3 Precision-Recall curves in link prediction

Most research on LP use ROC curves ([8, 12, 34, 35, 39]) or PR curves ([2, 41]) for evaluation, but for the reasons discussed in §2.3 we find PR curves to be more appropriate. PR curve shows the

performance of a classifier at various thresholds: at the left part of the curve are the high-certainty predictions where precision is higher, while at the right part of the curve are low-certainty predictions where recall grows at the expense of a lower precision. Through the PR curve one can see which classifier performs better at each threshold. The derived AUPR metric of the PR curve on the other hand determines which classifier performs better overall, when all thresholds are considered at the same time with the same importance. Due to this last point, we find the AUPR score to be sub-optimal for evaluating the applicability of results. Given the imbalance of the graphs used (see Table 1), a large part of the PR curve represents very low precisions. As recall grows precision can quickly reach levels unacceptable from a practical point of view. At this point one must consider which results are worth taking into account for evaluating performance. If we intend to achieve an applicable methodology we should focus on its performance where it matters, when a *reasonable* number of mistakes are being done. At extremely low precisions (*e.g.*, 0.01%) results are likely to be useless, and therefore should not be taken into account (certainly not with equal weight) into the evaluation. For this reason in §4 we propose a modified version of the AUPR measure with the goal of focusing on applicability.

4 Constrained AUC

The goal of classic binary classification is to fully discriminate two sets. This is idoneous when dealing with balanced or *typically* imbalanced domains, but becomes problematic for domains with class imbalances in the order of millions (*e.g.*, see Table 1). In this context, exhaustively discriminating the two classes becomes exceedingly complicated, which eventually renders a portion of the results obtained useless (see Figure1). Evaluation methodologies are unaware of the actual utility of each portion of the results, and combine the evidence provided by all results equally. Consequently, as the portion of results that is useless grows, the usefulness of the evaluation methodologies results decreases.

The LP problem is one where this disjunction between usefulness and classification performance takes place. As a solution we argue that the goal of LP is to produce high certainty and high utility predictions, instead of fully discriminating two sets. Indeed, LP does not need to classify most edges correctly in order to be useful, while trying to correctly classify all possible edges is a virtually impossible task due to the complexity of the problem. Instead, for the sake of making it useful for real world applications, LP should try to identify as many positive edges as possible, while keeping the number of false positives within an acceptable range.

To formally evaluate performance in terms of *usefulness* we first need to ground that subjective term. Since every domain, application and even user may have its own definition of it, we decide to define instead a minimal threshold which guarantees that all useful results are beyond it. If the hypothesis is accepted, all relevant results will be provided and accounted for, and utility, although not optimized, will be improved. The hypothesis we propose is as follows:

Hypothesis 1 *Given a link prediction process applied on a graph $G = (V, E)$, once the number of false positives is equal or larger than $|E|$, all further predictions become irrelevant.*

The idea behind this hypothesis is that, given a data set X , we will rarely accept any result which includes a number of mistakes as large as X itself. This is a conservative approach that may hold for most applications and domains.

Based on this hypothesis we propose the Constrained AUC score (CAUPR when applied to the PR curve) with the goal of evaluating LP scores based only on the predictions produced while keeping an acceptable number of mistakes (*i.e.*, less than the graph size). The CAUPR is analogous to the traditional AUPR measure, computing the AUC of the PR curve where the number of non-existing edges mistakenly accepted by the score (*i.e.*, false positives) is equal or lower than the total number of edges in the graph. Once the number of false positives is larger than the number of edges, the CAUPR for the rest of the curve equals 0. In practice, the CAUPR is a subset of the AUPR, starting from the high confidence predictions (left side of the PR curve) and ending when a given threshold is reached.

As an example, see Figure 3, where the PR curves of a LP score are shown for two different graphs. The vertical cut on each curve represents the location of the CAUPR threshold for each particular

data set and score, limiting the CAUPR to the area at the left of the threshold (coloured in grey), whereas the AUPR considers the whole curve.

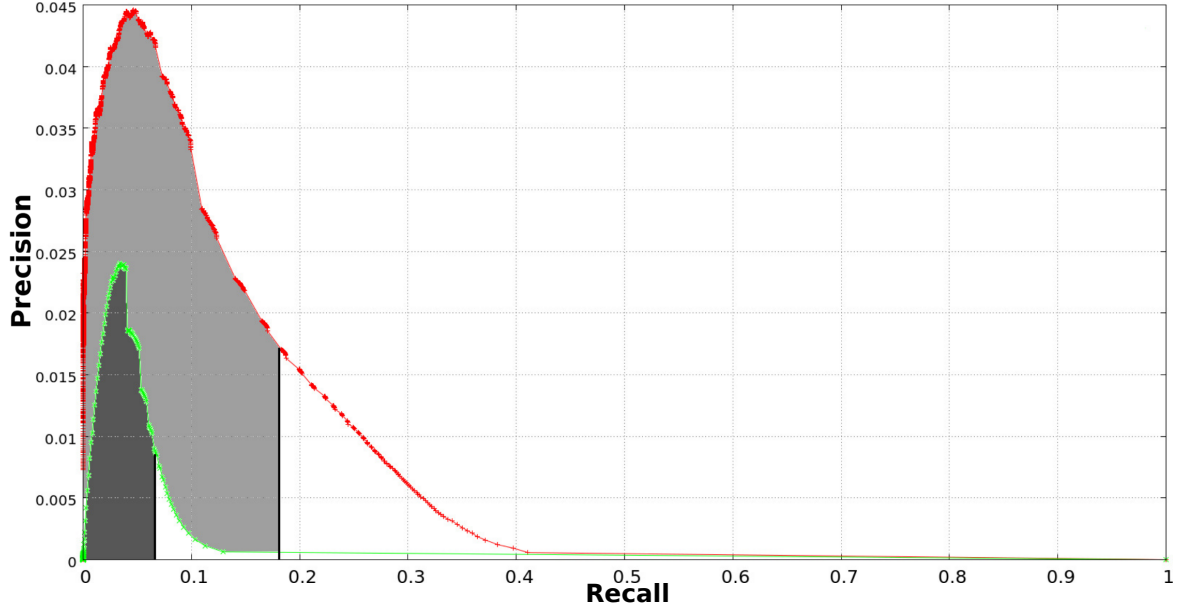


Figure 3: Precision-recall curve of the RA LP score on the Cyc graph (upper curve), and precision-recall curve of the same algorithm on the IMDb webgraph (lower curve). Grey area shows the respective CAUPR. The CAUPR threshold for the Cyc graph corresponds to a higher recall value (approximately 0.18) due to the higher precision obtained in this graph, despite Cyc being a smaller graph than IMDb. The CAUPR threshold for the IMDb graph corresponds to a recall value of approximately 0.06, thus by the time it finds 6% of all positive edges, the algorithm has made as many mistakes as edges in the IMDb graph.

4.1 CAUPR properties

The goal of the CAUPR performance measure is to avoid the evaluation of irrelevant parts of the PR curve. For that purpose CAUPR defines a threshold x at which a given algorithm is accepting more false positives than edges available in the graph. For PR curves, x indicates the maximum recall an algorithm can obtain before reaching the threshold, and splits the PR curve in two. The PR sub-curve in the recall interval $[0, x]$ will be the one CAUPR will take into account, while the PR sub-curve in the recall interval $[x, 1]$ will be the difference between the CAUPR and the AUPR. Significantly, x is in the interval $[0, 1]$. It can be zero, if the first $|E|$ predictions made by the LP algorithm are mistakes, but it can also be one, if all true positives are found before $|E|$ false positives are accepted. In this last case the CAUPR ignores nothing of the curve, and is equal to the AUPR. Consequently, the CAUPR and AUPR measures will only differ when the LP algorithms do not perform well enough (as defined by Hypothesis 1)

The LP scores that may be penalized by CAUPR in comparison with the AUPR are those which outperform their competitors on the *irrelevant* part of the curve. Since that area, if existent, is located at the right side of the curve, and since the PR curve is monotonically decreasing, the CAUPR will penalize the scores producing more horizontal PR curves. On the other hand, LP scores which make more accurate predictions at the beginning of the curve, when the number of false positives is still assumable, but which quickly lose precision (*i.e.*, those with a more vertical PR curve), will be the ones to benefit from the CAUPR. Consider the PR curves of Figure 4. The more horizontal curve (H) has a larger AUPR than the more vertical curve (V), but V outperforms H in CAUPR. This is due to the higher precision obtained by V at high confidence predictions, which causes a larger portion of V's AUPR to be considered by CAUPR. H, on the other hand, outperforms V at higher recall values, and mostly when the number of false positives is already larger than the graph itself, beyond the CAUPR threshold.

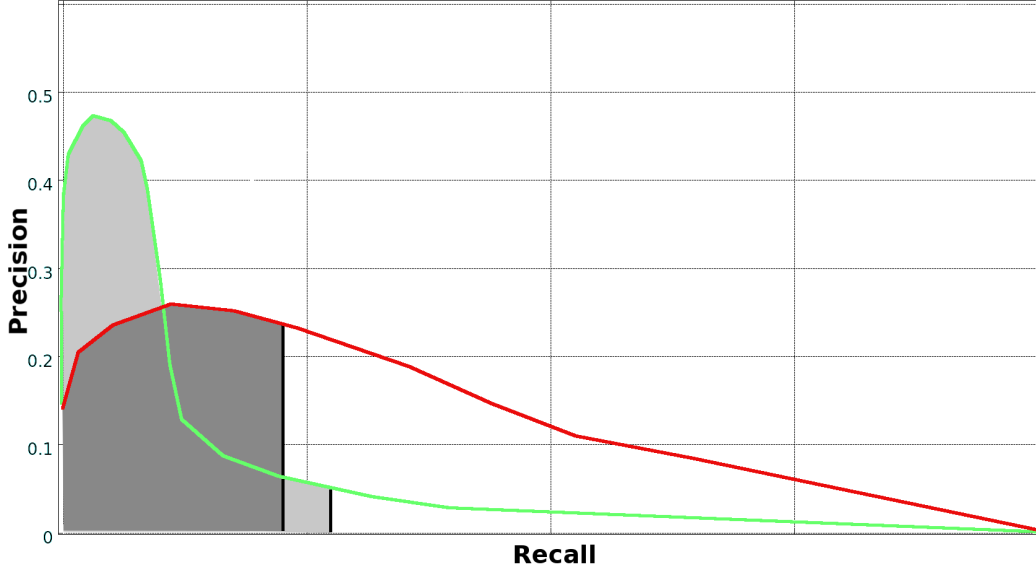


Figure 4: Illustration of how AUPR and CAUPR can produce contradictory comparisons due to different precision-recall curve shapes. Grey area represents CAUPR. In this example one curve has the larger AUPR, while the other has a larger CAUPR.

Another relevant feature provided by Hypothesis 1 is domain adaptation. By considering the number of edges in the graph as threshold, the CAUPR is affected by graph properties such as density and size (*i.e.*, large graphs will accept more mistakes than small ones, dense graphs will accept more mistakes than sparse ones). This is an interesting feature not found in the AUPR measure: AUPR evaluates the predictions done on a graph with N vertices and 1,000 edges and the predictions done on a graph with N vertices and 100,000 edges under the same conditions, as if these two problems were equally difficult. A clearly unrealistic assumption that complicates the interpretability of results. CAUPR, on the other hand, implicitly incorporates the size and density of the graph into the evaluation, allowing more concessions when they are acceptable by the domain, according to Hypothesis 1).

4.2 CAUPR impact

To empirically evaluate the impact of the CAUPR measure we use three well known LP algorithms: Common Neighbours (CN, [40]), Adamic Adar (AA, [1]) and Resource Allocation (RA, [51]). For each of those algorithms we compute their AUPR and CAUPR when applied to the nine graphs described in Table 1. The algorithm obtaining the best AUPR on each graph (*i.e.*, the algorithm of reference) and its corresponding AUPR and CAUPR values can be seen in the first four columns of Table 3. Each of the three LP algorithms obtains the best AUPR score on at least one of the nine graphs.

For each of the nine graphs we compare the results obtained by the algorithm of reference with the results obtained by the remaining two algorithms. We show the AUPR and CAUPR values of these two algorithms as a percentage of the values of reference, so that for example, for the WordNet graph, the AUPR of the AA algorithm is shown to represent a 46,3% of the AUPR of the algorithm of reference (see 2nd row, 6th column of Table 3). By also showing the percentage relation for the CAUPR measure, we can see the relative difference between both measuring methods. Using the same example, since the CAUPR of the AA algorithm is 40,7% of the CAUPR of RA, the difference is -5,6% points. Thus, the difference in performance between the AA and RA algorithms is 5,6% larger according to the CAUPR measure than for the AUPR measure.

The *Impact* column of Table 3 shows the changes in performance according to CAUPR, and shows how this measure may provide a relevant variation in the evaluation of LP for graphs with 100,000 vertices or more. Significantly, the variation provided by the CAUPR measure does not benefit any of the three algorithms: all of them have positive and negative differences. Results also indicate that

Table 3: For 9 graphs, 2nd to 4th columns show best AUPR obtained by one of CN, AA and RA algorithms (*i.e.*, algorithm of reference), and the CAUPR obtained by that same algorithm. Remaining columns show the AUPR and CAUPR obtained by the other two algorithms as a percentage of the results obtained by the algorithm of reference. *Impact* column shows the difference of the percentages, summarizing how CAUPR alters the comparison between the algorithm of reference and the other two LP algorithms.

Graph	Ref. Alg.	AUPR of ref.	CAUPR of ref.	Alg.	AUPR as % of ref.	CAUPR as % of ref.	Impact
WN	RA	0,0487	0,0383	AA	46,3%	40,7%	-5,6%
				CN	13,1%	12,7%	-0,4%
Cyc	RA	0,0076	0,0058	AA	79,5%	83,5%	+4,0%
				CN	19,0%	23,2%	+4,2%
WebND	CN	0,3185	0,3158	RA	66,4%	65,8%	-0,6%
				AA	99,4%	99,0%	-0,4%
WebSB	RA	0,0549	0,0460	AA	40,3%	40,8%	+0,5%
				CN	30,3%	32,7%	+2,4%
WebGL	RA	0,1003	0,0921	AA	89,2%	88,6%	-0,6%
				CN	62,0%	60,8%	-1,2%
IMDb	RA	0,0015	0,0011	AA	62,7%	78,2%	+15,5%
				CN	43,2%	54,8%	+11,6%
Hudong	CN	0,0074	0,0072	RA	30,0%	21,8%	-8,2%
				AA	74,6%	69,7%	-4,9%
Baidu	RA	0,0031	0,0016	AA	92,5%	94,6%	+2,1%
				CN	57,1%	66,6%	+9,5%
DBp	AA	0,0005	0,0003	RA	28,5%	351,8%	+323,3%
				CN	67,8%	73,8%	+6,0%

variations tend to increase with the graph size, since larger graphs typically have larger imbalances, which often imply a lower recall threshold for the CAUPR measure. Clearly, having a lower recall threshold makes it easier (though not necessary) for the CAUPR and AUPR measures to differ.

Table 4 shows the recall thresholds for the CAUPR measure of every graph and algorithm tested. This table gives a measure of the portion of the PR curve that is being disregarded by the CAUPR method. A threshold of 0.1 implies that 90% of the curve is outside the CAUPR range, and therefore not considered in the CAUPR evaluation. The impact of the CAUPR measure, powered by the class imbalance, is highlighted by the fact that for five of the nine graphs a majority of the curve is irrelevant according to Hypothesis 1.

One remarkable result to be considered is the variation on the largest graph used, the DBpedia graph. In this domain, the AA algorithm outperforms the rest according to the AUPR measure. However, according to the CAUPR measure the RA algorithm is best instead, with a three times larger CAUPR. To analyze these results let's first consider the DBpedia graph, which has the largest class imbalance of those graphs here considered, with more than 2 million negative edges for each positive one. As shown in Table 4, RA performs very well on the DBpedia graph, reaching a recall of 26% before the threshold of mistakes is attained. Comparably, AA retrieves only a 5% of all positive edges by the time it reaches the threshold. Nevertheless, AA seems to outperform RA for the portion of the curve beyond the threshold, thus obtaining a higher AUPR value. An example of PR curves with this kind of behaviour are illustrated in Figure 4, and are a showcase of the relevance of the proposed CAUPR measure.

5 Related Work

To the best of our knowledge, there are no previous proposals on how to adapt standard evaluation methods (*i.e.*, PR curves) to the particularities of large-scale LP. Similar solutions to our own (that of using a sub-part of the PR curve) have been previously considered for ROC curves on other contexts, particularly in the domain of diagnostic medicine, where several authors have considered the possibility of using only a partial ROC curve [36, 52]. In this field, the metrics used to cut the ROC

Table 4: For 9 different graphs, CAUPR threshold showing at which recall value the number of mistakes is larger than the graph size. AUC beyond this recall value is not considered by the CAUPR measure.

Graph	RA CAUPR Recall Threshold	AA CAUPR Recall Threshold	CN CAUPR Recall Threshold
WN	0.539892	0.276364	0.147471
Cyc	0.183218	0.144625	0.092161
WebND	0.673279	0.636334	0.558378
WebSB	0.509361	0.234861	0.159888
WebGL	0.619519	0.564522	0.474023
IMDb	0.073010	0.048817	0.034015
Hudong	0.073233	0.081013	0.069519
Baidu	0.096464	0.095531	0.076136
DBpedia	0.267792	0.055546	0.039784

curve are often clinical relevance and clinical application. Rather differently, our proposal constrains the PR curve based on a domain agnostic measure: the input data set imbalance. The methodology we propose is thus applicable to virtually any LP evaluation problem, regardless of the data origin.

6 Conclusions

Two of the most disturbing features of real world graphs for the evaluation of LP algorithms are their size and scale-free topology. Medium sized graphs (*e.g.*, up to a few million vertices) are hard to compute through exhaustive algorithms, and have motivated the design of graph-specific parallel models (*e.g.*, [37]). However, this same size can also simplify certain data mining steps, such as assessing test set construction as an independent dataset. This assessment, which is often implemented through 10-fold cross-validation, is actually avoidable in medium and larger graphs, since the size of a random 10% split (*e.g.*, hundreds of thousands of vertices) already guarantees the construction of a stable sample (see Table 2). These results were consistent for graphs between 100,000 vertices and 17 million vertices, and can be extended to any graph larger than those. Avoiding cross validation can entail significant savings in terms of computational resources, allowing one to reduce the cost of every performance evaluation process by a factor of ten (assuming we were to use 10-fold cross-validation).

The second graph feature which is particularly relevant for evaluation is related with the scale-free topology of many real world graphs. Since LP can be reduced to a binary classification problem, where one tries to separate a positive class (the edges missing from the graph which should be added) from a negative class (the edges missing which should not be added), the scale-free topology implies a huge imbalance between both classes. In fact, imbalance reaches a degree rarely found in the bibliography, where, for every positive instance, there are tens of thousands or even millions of negative ones. Significantly, imbalance becomes larger as graphs do, making this an issue for current and future graph mining applications.

One side effect of huge class imbalance relates with the evaluation methodology being used. Binary classification problems are often evaluated through ROC curves, which plot TPR against FPR. FPR is however an uninformative performance scale in highly imbalanced domains, as most of the curve implies a huge amount of false positives (see illustrative Figure 1). For LP in medium or large graphs, PR curves provide a much more realistic performance measure, since these curves plot precision against recall, directly displaying the number of false positives being done.

Unfortunately, using the PR curve does not guarantee the utility or correct interpretability of results, particularly if using the associated AUC measure. The large imbalance found in LP for large graphs often results in small precision values, which only get worse as recall grows. As a result, a significant part of the AUPR measure may correspond to predictions found at the cost of an unassumable amount of mistakes, and thus poorly represent applicable performance. To tackle this problem, we define a constrained version of the AUPR measure, CAUPR, by setting a conservative threshold for what number of mistakes are assumable. This threshold is based on the graph size

(i.e., we can accept as many false positives as edges in the graph), which provides several interesting properties. For example, the CAUPR may be equal to the AUPR, if performance is good enough, but it can also be zero if performance is very poor. Also, the CAUPR adapts to graph size and density, being more flexible when the domain allows. Nevertheless, the use of the CAUPR measure requires of the acceptance of Hypothesis 1, which should be considered on a case by case basis.

Our empirical comparison between the AUPR and the CAUPR measures shows significant variances between both performance metrics, which tend to increase with graph size. On the evaluation of three LP algorithms, AUPR and CAUPR differ on which is the best one when applied to the largest graph computed (DBpedia, 17 million vertices), showcasing the relevance of the AUPR performance metric for LP evaluation on large and highly imbalanced graphs.

Acknowledgements

This work is partially supported by the Joint Study Agreement no. W156463 under the IBM/BSC Deep Learning Center agreement, by the Spanish Government through Programa Severo Ochoa (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project and by the Generalitat de Catalunya (contracts 2014-SGR-1051).

References

- [1] Lada A Adamic and Eytan Adar. “Friends and neighbors on the web”. In: *Social networks* 25.3 (2003), pp. 211–230.
- [2] Charu C. Aggarwal, Yan Xie, and Philip S. Yu. “A framework for dynamic link prediction in heterogeneous networks”. In: *Statistical Analysis and Data Mining* (2013). ISSN: 1932-1872. DOI: 10.1002/sam.11198.
- [3] Edoardo M Airolti et al. “Mixed membership stochastic block models for relational data with application to protein-protein interactions”. In: *Proceedings of the international biometrics society annual meeting*. 2006, p. I5.
- [4] Mohammad Al Hasan et al. “Link prediction using supervised learning”. In: *SDM’06: Workshop on Link Analysis, Counter-terrorism and Security*. 2006.
- [5] Réka Albert, Hawoong Jeong, and Albert-László Barabási. “Internet: Diameter of the world-wide web”. In: *Nature* 401.6749 (1999), pp. 130–131.
- [6] Robin Burke. “Hybrid recommender systems: Survey and experiments”. In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.
- [7] Nitesh V Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.
- [8] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. “Hierarchical structure and the prediction of missing links in networks”. In: *Nature* 453.7191 (2008), pp. 98–101.
- [9] Diane J Cook et al. “Structural mining of molecular biology data”. In: *Engineering in Medicine and Biology Magazine, IEEE* 20.4 (2001), pp. 67–74.
- [10] Jesse Davis and Mark Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 233–240.
- [11] Tom Fawcett. “ROC graphs: Notes and practical considerations for researchers”. In: (2004).
- [12] Michael Fire et al. “Link prediction in social networks using computationally efficient topological features”. In: *2011 IEEE 3rd international conference on social computing*. 2011, pp. 73–80.
- [13] D Garcia-Gasulla. “Link Prediction in Large Directed Graphs”. PhD thesis. Universitat Politècnica de Catalunya - Barcelona TECH, 2015.
- [14] D. Garcia-Gasulla and U. Cortés. “Link Prediction in Very Large Directed Graphs: Exploiting Hierarchical Properties in Parallel”. In: *3rd Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data - 11th Extended Semantic Web Conference* (2014).

- [15] Dario Garcia-Gasulla et al. “Evaluating Link Prediction on Large Graphs”. In: *Proceedings of the 18th International Conference of the Catalan Association of Artificial Intelligence*. Vol. 277. Artificial Intelligence Research and Development. IOS Press, 2015, pp. 90–99.
- [16] Lise Getoor and Christopher P Diehl. “Link mining: a survey”. In: *ACM SIGKDD Explorations Newsletter* 7.2 (2005), pp. 3–12.
- [17] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.
- [18] Roger Guimera and Luis A Nunes Amaral. “Functional cartography of complex metabolic networks”. In: *Nature* 433.7028 (2005), pp. 895–900.
- [19] Roger Guimerà and Marta Sales-Pardo. “Missing and spurious interactions and the reconstruction of complex networks”. In: *Proceedings of the National Academy of Sciences* 106.52 (2009), pp. 22073–22078.
- [20] Haibo He and E.A. Garcia. “Learning from Imbalanced Data”. In: *Knowledge and Data Engineering, IEEE Transactions on* 21.9 (2009), pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239.
- [21] Lawrence B Holder and Diane J Cook. “Graph-Based Data Mining.” In: *Encyclopedia of data warehousing and mining* 2 (2009), pp. 943–949.
- [22] Zan Huang, Xin Li, and Hsinchun Chen. “Link prediction approach to collaborative filtering”. In: *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2005, pp. 141–142.
- [23] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. “An apriori-based algorithm for mining frequent substructures from graph data”. In: *Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 13–23.
- [24] Nathalie Japkowicz and Shaju Stephen. “The class imbalance problem: A systematic study”. In: *Intelligent data analysis* 6.5 (2002), pp. 429–449.
- [25] Brian Karrer and Mark EJ Newman. “Stochastic blockmodels and community structure in networks”. In: *Physical Review E* 83.1 (2011), p. 016107.
- [26] Ashraf Khalil and Yong Liu. “Experiments with PageRank computation”. In: *Indiana University, Department Computer Science*. URL: <http://www.cs.indiana.edu/~akhalil/Papers/pageRank.pdf> (2004).
- [27] Jon Kleinberg. “Authoritative sources in a hyperlinked environment”. In: *Journal of the ACM (JACM)* 46.5 (1999), pp. 604–632.
- [28] Valdis E Krebs. “Mapping networks of terrorist cells”. In: *Connections* 24.3 (2002), pp. 43–52.
- [29] Jérôme Kunegis. “KONECT: the Koblenz network collection”. In: *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee. 2013, pp. 1343–1350.
- [30] Jens Lehmann et al. “DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia”. In: *Semantic Web Journal* (2014).
- [31] Douglas B Lenat. “CYC: A Large-Scale Investment in Knowledge Infrastructure”. In: *Communications of the ACM* 38 (1995), pp. 33–38.
- [32] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. “New perspectives and methods in link prediction”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 243–252.
- [33] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. “Exploratory undersampling for class-imbalance learning”. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39.2 (2009), pp. 539–550.
- [34] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. “Similarity index based on local paths for link prediction of complex networks”. In: *Physical Review E* 80.4 (2009), p. 046122.
- [35] Linyuan Lü and Tao Zhou. “Link prediction in complex networks: A survey”. In: *Physica A: Statistical Mechanics and its Applications* 390.6 (2011), pp. 1150–1170.
- [36] Hua Ma et al. “On use of partial area under the ROC curve for evaluation of diagnostic performance”. In: *Statistics in medicine* 32.20 (2013), pp. 3449–3458.

- [37] Grzegorz Malewicz et al. “Pregel: a system for large-scale graph processing”. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, pp. 135–146.
- [38] Amin Mantrach et al. “The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.6 (2010), pp. 1112–1126.
- [39] Tsuyoshi Murata and Sakiko Moriyasu. “Link prediction based on structural properties of online social networks”. In: *New Generation Computing* 26.3 (2008), pp. 245–257.
- [40] Mark EJ Newman. “Clustering and preferential attachment in growing networks”. In: *Physical Review E* 64.2 (2001), pp. 251021–251024.
- [41] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “A three-way model for collective learning on multi-relational data”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 809–816.
- [42] Lawrence Page et al. “The PageRank citation ranking: bringing order to the web.” In: (1999).
- [43] Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. “Towards time-aware link prediction in evolving social networks”. In: *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*. ACM. 2009, p. 9.
- [44] Koji Ueno and Toyotaro Suzumura. “Highly scalable graph search for the graph500 benchmark”. In: *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*. 2012, pp. 149–160.
- [45] Christian Von-Mering et al. “Comparative assessment of large-scale data sets of protein–protein interactions”. In: *Nature* 417.6887 (2002), pp. 399–403.
- [46] Takashi Washio and Hiroshi Motoda. “State of the art of graph-based data mining”. In: *ACM SIGKDD Explorations Newsletter* 5.1 (2003), pp. 59–68.
- [47] Mike Wasikowski and Xue Wen Chen. “Combating the small sample class imbalance problem using feature selection”. In: *Knowledge and Data Engineering, IEEE Transactions on* 22.10 (2010), pp. 1388–1400.
- [48] Masaru Watanabe and Toyotaro Suzumura. “How social network is evolving? A preliminary study on billion-scale twitter network”. In: *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee. 2013, pp. 531–534.
- [49] Gary M Weiss. “Mining with rarity: a unifying framework”. In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 7–19.
- [50] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. “Evaluating link prediction methods”. In: *Knowledge and Information Systems* (2014), pp. 1–32.
- [51] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. “Predicting missing links via local information”. In: *The European Physical Journal B* 71.4 (2009), pp. 623–630.
- [52] Xiao-Hua Zhou, Donna K McClish, and Nancy A Obuchowski. *Statistical methods in diagnostic medicine*. Vol. 569. John Wiley & Sons, 2009.